ALGORITHMS FOR HIGH-SPEED UNIVERSAL NOISELESS CODING

Robert F. Rice*
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Pen-Shu Yeh**
Goddard Space Flight Center
Greenbelt, Maryland

Warner Miller**
Goddard Space Flight Center
Greenbelt, Maryland

Abstract

This paper provides the basic algorithmic definitions and performance characterizations for a high-performance adaptive noiseless (lossless) 'coding module" which is currently under separate developments as single-chip microelectronic circuits at two NASA centers. Laboratory tests of one of these implementations recently demonstrated coding rates of up to 900 Mbits/s. Operation of a companion "decoding module" can operate at up to half the coder's rate. The functionality provided by these modules should be applicable to most of NASA'S science data.

The hardware modules incorporate a powerful adaptive noiseless coder for "Standard Form" Data Sources (i.e., sources whose symbols can be represented by uncorrelated non-negative integers where the smaller integers are more likely than the larger ones). Performance close to data entropies can be expected over a "Dynamic Range" of from 1,5 to 12-15 bits/sample (depending on the implementation).

This is accomplished by adaptively choosing the best of many "Huffman Equivalent" codes to use on each block of 1-1 b samples. Because of the extreme simplicity of these codes, no table lookups are actually required in an implementation, thus leading to the expected very high data rate capabilities already noted. The "coding module" can be used directly on data which has been "pre-processed" to exhibit the characteristics of a Standard Form Source. Alternatively, a built-in Predictive Pre-processor can be used where applicable, ? his built-in Preprocessor includes the familiar one-dimensional predictor followed by a function which maps the prediction error sequences into the desired standard form. Additionally, an External prediction can be substituted if desired (e.g, fOr two-dimensional applications), further extending the module's generality.

I. Introduction

References 1-4 provide the development and analysis of some practical adaptive techniques for efficient noiseless (lossless) coding of a broad class of data sources. These have been applied, In various forms, to numerous applications

Those functions and algorithms most desirable for incorporation In a 'coding module" which could be implemented using current custom VLSI capabilities were presented at the first NASA Data Compression Workshop at Snowbird, Utah, in 1988.[5] A workshop committee recommended that NASA should proceed and implement this "coding module." Since then, both the Jet Propulsion Laboratory (JPL) and the Microelectronics Research Center (MRC) at the University of New Mexico have implemented first-generation single-chip CMOS VLSI coding modules.[6-9] The MRC 1.0 µm coding chip was successfully tested under laboratory conditions at up to 900 Mbits/s.[7-9] A companion MRC "decoding module" is designed to run at up to half the maximum rate of the coding module. These first-Generation MRC chips are now available commercially from Advanced Hardware Architectures in Moscow, Idaho.[10]

Both the MRC and JPL developments are nearing completion of second-generation space-qualified versions for the coding modules[11-14]. It is anticipated that the high performance functionality of these modules, first- and second-generation, can serve most of NASA's science data needs where a lossless representation is appropriate.

The intent of this paper is to provide a concise description of the basic algorithmic and performance characteristics which are embodied In the coding modules A more general algorithmic development can be found In Ref. 15 along with some application notes. Observe that the actual implementations have diverged slightly from the definitions provided here and from each other. Most of these subtleties will be discussed

*Member of Technical Staff
**Engineer in Electronic Systems Branch

II. The Coding Module

A functional block diagram of a general-purpose lossless "coding module" is shown in Fig. 1, A variation in Rice's original notation (of subscripting the Greek letter ψ) will be used to name various coding operations. Subsequent sections will quickly converge to more specific definitions that relate to the VLSI modules being implemented,

The input to this coding module

$$\tilde{X}^n = x_1\, x_2 \ldots x_J \qquad (1)$$

is a J sample block of n bit samples, $\tilde{Y}$ is a priori or Side Information that might help in the coding process

The overall unspecific process of representing $\tilde{X}^n$ is named PSI?+ so that the actual coded result is

$$PSI? + [\tilde{X}^n, \tilde{Y}]$$

As Fig. 1 shows, the coding process is split into two independent steps discussed below.

Step 1

A Reversible Pre-processor is a process designed to convert the source represented by $\tilde{X}^n$ sequences (and $\tilde{Y}$) into a close approximation to a STANDARD FORM Data Source, represented by $\tilde{\delta}^n$ sequences. This process usually Includes a decorrelation procedure (prediction).

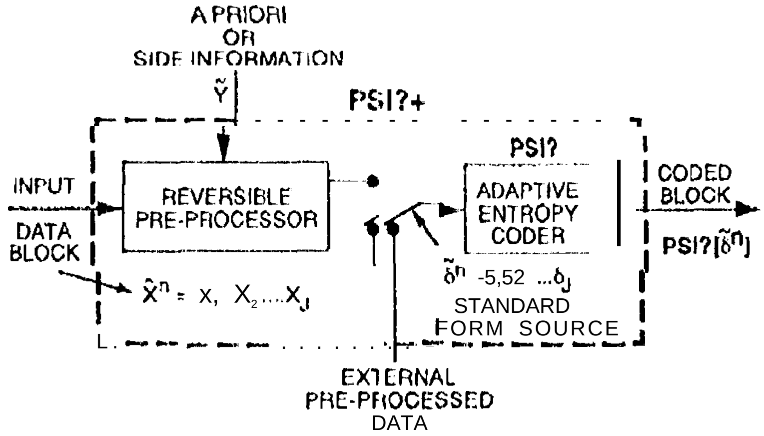

Fig. 1. **General-Purpose Noiseless** Coding **Module Block** Diagram

The pre-processor converts each $\tilde{X}^n$ (and corresponding $\tilde{Y}$, If any) Into

$$\tilde{\delta}^{n'} = \delta_1 \delta_2 \ldots \delta_J \quad (2)$$

a $J \geq 1$ sample sequence of n' bit samples. Usually $n = n'$, and we will henceforth assume that here,

Standard Form Source. Specifically,

a) Samples of $\tilde{\delta}^n$ are the non-negative integers $0, 1, 2, \ldots, q$ (3)

b) Samples of $\tilde{\delta}^n$ are independent (4)

c) With $P_i = Pr[\delta_j = I]$, the probabilities are ordered so that the smaller integers occur more frequently, i.e.,

$$p_0 \geq p_1 \geq p_2 \geq \ldots \quad (5)$$

d) The source entropy is given as

$$H_\delta = \sum_i p_i \log_2 p_i \text{ bits/sample} \quad (6)$$

The best pre-processor will meet these conditions and produce the lowest H_δ.

Step 2

An adaptive entropy coder, named PSI? for now, efficiently represents (Standard Source) pre-processed $\tilde{\delta}^n$ sequences with the code-d result

$$PSI?[\tilde{\delta}^n] = Psi?+[\tilde{X}^n, \tilde{Y}]$$

This entropy coder is independent of the pre-processor. The pre-processor's goal is to achieve performance that remains close to H_δ as It varies with time.

Note that the coding module In Fig, 1 allows for coder PSI? to be used directly on externally supplied pre-processed data.

Prelude to the Details

A general form of an Adaptive Entropy coder (designed to efficiently represent Standard Data Sources), which chooses from multiple algorithm options on a block-by-block basis, will be identified in the next section. Specific sets of such code options will be defined and Incorporated In this structure as a parametrically defined adaptive coder. In doing so, the unspecific "PSI?" will be turned Into a specific coder called "PSIss."

Finally, the specific parameters of PSIss that are used in current VLSI implementations will be identified.

III. Adaptive Entropy Coder for the Standard Source

PSI? of Fig 1 represents the general-purpose adaptive coder called PSI1 1 in Refs. 2-4 and 15. Basically, such a coder chooses one of a set of Code Options (coding algorithms) to use to represent an incoming block of "pre-processed" data samples. A unique binary identifier precedes a coded block to tell a decoder which decoding algorithm to use, The following discussion will identify specific code options.

Code Options

Backup. When no coding of any form is performed on the data, we call this PSIbu

$$PSIbu[\tilde{\delta}^n] = \tilde{\delta}^n \quad (7)$$

This representation Is used In an adaptive coder when all other available code options fail to compress $\tilde{\delta}^n$. References 7-9 call this the "default" option,

The Fundamental Sequence Code. Recall that the pre-processed samples of $\tilde{\delta}^n$ are the non-negative integers $I \geq 0$. A variable length "Fundamental Sequence Code, fs", is defined for each I as follows

$$fs[i] = \overbrace{000\ldots\ldots000}^{i \text{ zeroes}}1 \quad \text{for } i \geq 0 \quad (8)$$

That is, simply append a 1 to the end of a sequence o' i zeroes.

The "Fundamental Sequence" itself Is the application of $fs[\cdot]$ to all the samples of $\tilde{\delta}^n$. Following Rice's notation,*

$$PSI1[\tilde{\delta}^n] = fs[\delta_1] * fs[\delta_2] * \ldots fs[\delta_J] \quad (9)$$

is the Fundamental Sequence, This defines Code Option, PSI1

Split-Sample Modes. The code option definitions here are basically to "split" off the k least significant bits of each δ^n sample and send them separately. The remaining n-k most significant bit samples are then crxforf u$ing F'SII. Specifically, with

$$\delta^n = \delta_1 \delta_2 \ldots \delta_J$$

Let

$$\tilde{M}^{n,k} = m_1\ m_2\ \ldots\ m_J \quad (10)$$

be the sequence of all the n-k most significant bit samples of δ^n and lot

$$\tilde{L}_k = \ell sb_1 * \ell sb_2 * \ldots \ell sb_J \quad (11)$$

denote the corresponding sequence of all the k-bit least significant bit samples of $\tilde{\delta}^n$.

That is

$$\delta_i = m_i * \ell sb_i \quad (12)$$

The "Split. Sample" Mode Code Option PSI I ,k is defined by

$$PSI1_{,k}[\tilde{\delta}^n] = PSI1[\tilde{M}^{n,k}] * \tilde{L}_k \quad (13)$$

Note that k = 0 Isa special case where

$$PSI1,0 = PSI1 \quad (14)$$

and when k = n

$$PSI1,n = PSIbu \quad (15)$$

The Individual Code Words. Note that the individual code word assigned to δ_i in (12) when code option PSI1,k Is applied Is given as

$$fs[m_i] * \ell sb_i \quad (16)$$

That Is, the Fundamental Sequence Code, $fs[\cdot]$ in (9), is applied to the most significant n-k bits of δ_i, followed by tho least significant k bits of δ_i. From this description, it should be easier to soo that the only variable-length-code operation ever required Is the application of $fs[\cdot]$, since the "ℓsbs" can simply be shifted out, and $fs[\cdot]$ can be implemented without any table lookups.

Performance of the Individual PSI1,k Options. Under certain familiar assumptions about the type of data source (these assumptions will be described in a later section), the individual ' variable length codes" represented by (16) were shown by Yeh[13,14] to be equivalent to Huffman Codes. (17)

Thus they are not only extremely simple, they are optimum too.

But even more important for their application In an adaptive coder, the entropy where PSI 1,k achieves its best performance is at

$$\bar{H}_\delta^k \approx k + 2 \text{ bits/sample} \quad (18)$$

and performance remains close to $\bar{H}_\delta^k$ over a range of about ±0.5 bit/sample. Thus there is at least one PSI1,k option that should provide efficient coding for any

$$\bar{H}_\delta > 1.5 \text{ bits/sample} \quad (19)$$

Such conclusions can also be drawn directly from simulations using a broad range of data sources.

Split-Sample Adaptive Coder, PSIss

We can now use the Split-Sample Modes described above to replace the general coder PSI? in Fig. 1 with a specific class of parametrically defined adaptive coders, named PSIss. PSIss is based on the following parameters:

J = block size ≥ 1

n . Input Bits/Sample (20)

N = number of Code Options

λ ≥ 1 (Dynamic Range Parameter)

A functional block diagram is shown in Fig. 2.

1 he representation of J sample $\tilde{\delta}^n$, using an N option i 'S1ss with parameter λ ≥ 1, is given by

$$PSIss[\tilde{\delta}^n] = ID(id) * PSI1, k(id)[\tilde{\delta}^n] \quad (21)$$

where

$$id = 0, 1, 2, \ldots, N-1 \quad (22)$$

is the integer value of a coder identifier for the options used, and ID(id) is its standard binary representation, requiring**

$$\lceil \log_2 N \rceil \text{ bits} \quad (23)$$

F-HI ,k(id) is the Split-Sample option specified by id, who: e

$$k(id) = \begin{cases} n \text{ for } id = N-1 \\ \lambda - 1 + id \text{ Otherwise} \end{cases} \quad (24)$$

for parameter λ ≥ 1 By (15) and (24) the last option is PSIbu in (7).

* An asterisk, ●, is used to emphasize the concatenation of sequences.

** ⌈z⌉ is the smallest integer, greater than or equal to z.

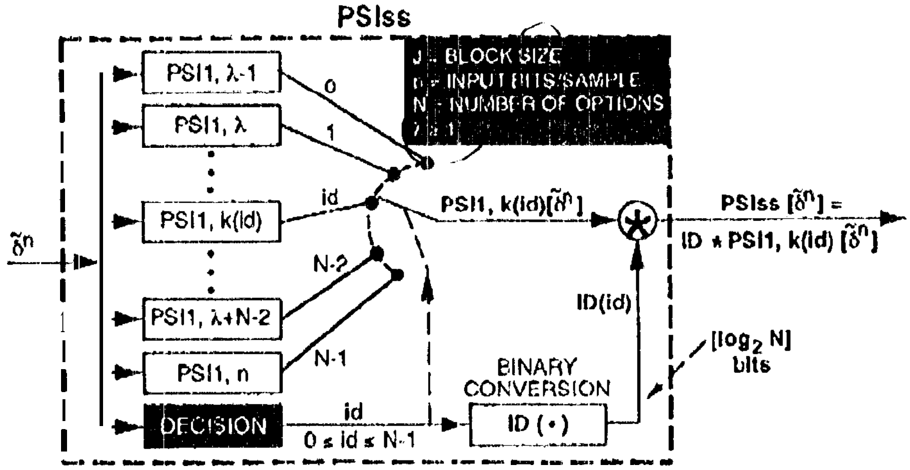

Fig. **2.** Parametric Split-Sample Adaptive Coder **Functional Block** Diagram

Dynamic Range. Except for limiting cases, the range of entropies where PSIss can be expected to efficiently represent preprocessed $\tilde{\delta}^n$ sequences has been shown to be closely specified by

$$\lambda + 0.5 \le \tilde{H}_\delta \le \min \begin{cases} n \\ \lambda + N - 0.5 \end{cases} \tag{25}$$

A close look at this expression shows that **each** increase in λ **moves the Dynamic Range** of **efficient performance upwards by 1 bit/sample.**

Performance Graph. A graph of typical performance for PSIss with $N = 12, \lambda = 1, n = 14$ and $J = 16$ is shown in Fig. 3.

Choosing the Right Option. The optimum criterion for selecting the best option to use to represent $\tilde{\delta}^n$ is to simply choose the one that produces the shortest coded sequence. That is,

choose $k = k^*$ if *

$$\mathcal{L}(PSI1,k^*[\tilde{\delta}^n]) = \min_k \{\mathcal{L}(PSI1,k[\tilde{\delta}^n])\} \tag{26}$$

Now, letting

$$F_k = \mathcal{L}(PSI1[\tilde{M}^{n,k}])$$

we have from (12)

$$\mathcal{L}(PSI1,k[\tilde{\delta}^n]) = F_k + Jk \tag{27}$$

and from (8) - (10)

$$F_k = \sum_{i=1}^{J} m_i + J \tag{28}$$

(i.e., the sum of the most significant n-k bit samples plus the block size),

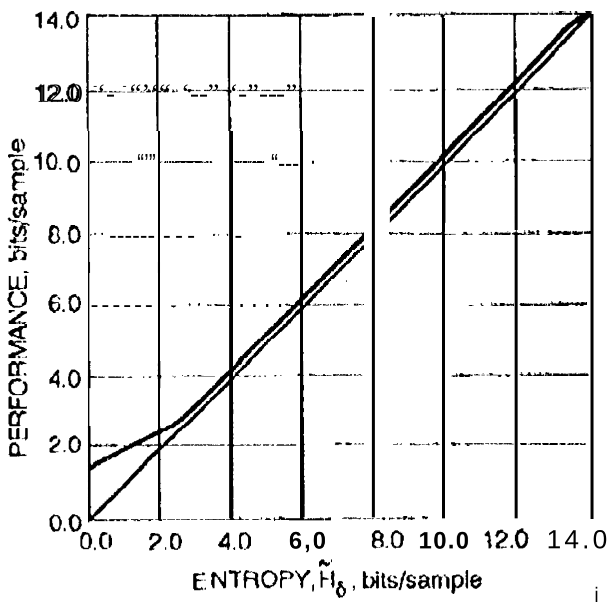

Fig. 3. **PSIss Average Performance** for $N = 12$, $n = 14, \lambda = 1, J = 16$

Thus (27) and (28) can simplify the decision making in (26) without actually coding the data. But this can be further simplified.

By taking advantage of the randomness in the least significant bits., the expected value of F_k can be related to F_0 by[4,15]

$$E(F_k|F_0) = 2^{-k}F_0 + \frac{J}{2}(1-2^{-k}) \tag{29}$$

which we use as an estimate in (27'), We have

$$Y_{1,k}(\tilde{\delta}^n) = 2^{-k}F_0 + \frac{J}{2}(1-2^{-k}) + Jk$$

$$\approx \mathcal{L}(PSI1,k[\tilde{\delta}^n]) \tag{30}$$

We can then choose $k = k^*$ if

$$Y_{1,k^*}(\tilde{\delta}^n) = \min_k \{Y_{1,k}(\tilde{\delta}^n)\} \tag{31}$$

* $\mathcal{L}(\tilde{Z})$ = length of sequence $\tilde{Z}$ in bits

and this leads to distinct decision regions based solely on F_0 (which by (28) can be determined by adding up the original samples since $m_i = \delta_i$). The boundaries to adjacent PSI1,k decision regions are given by

$$F_0 = \frac{J}{2} + J\,(2^{k+1})\text{ bits} \qquad (32)$$

Any PSI1,k option will generate more bits than PSIbu when

$$F_0 > \frac{J}{2}\left[\,(n-k)\,2^{k+1} + 1 - 2^{k}\right]\text{bits} \qquad (33)$$

A sample table of decision regions is shown in Table 1 for an N = 8 optics PSIss.

Note that if the largest value of k used is $k = \ell$, then PSI1,ℓ will generate more bits than PSIbu when

$$F_\ell > (n-\ell)\,J\text{ bits} \qquad (34)$$

which may be a simpler test than (33) provides, in some cases.

PSIss Implementation Parameters

The primary PSIss parameters used in the first-generation VLSI implementations were as follows:[6-9]

a) J = block size of $\tilde{\delta}n$ = 16

b) n = supported quantization of $\tilde{\delta}^n$ samples
= 12 for JPL chips
= 4, 6, ..., 14 for MRC chips

c) N = number of code options
= 11 for JPL chip
= 12 for MRC chip

d) λ = Dynamic Range Parameter = 1 = Starting option Parameter

Some Subtle Differences. By Eq. 23, the number of fixed identifier bits does not need to be more than $\lceil\log_2 N\rceil$ bits. A coder which represents 8-bit data has no need for more than 8 code options and thus should need no more than a 3-bit identifier. The second-generation MRC design will recognize

Table 1. Decision Regions for an N = 8 Option PSIss

CODE OPTION	F_0 REGION IN BITS
PSI1,0	$F_0 \le 5J/2$
PSI1,1	$5J/2 < F_0 \le 9J/2$
PSI1,2	$9J/2 < F_0 \le 17J/2$
PSI1,3	$17J/2 < F_0 \le 33J/2$
PSI1,4	$33J/2 < F_0 \le 65J/2$
PSI1,5	$65J/2 < F_0 \le 129J/2$
PSI1,6	$129J/2 < F_0 \le (128n-831)J/2$
PSIbu	$(128n-831)J/2 < F_0$

this situation. However, JPL's first- and second-generation chips and MRC's first generation chip fixed the number of identifier bits at four,

In the JPL case, the first.generation chip only supported input quantization of 12 bits/sample, so this was not an issue. A second design now provides support for data with quantization down to 8 bits/sample. But in doing so the number of code options is still fixed at 11, yielding 4 bite for identifiers.

The JPL support for n < 12 is provided by treating data of lesser quantization as right-justifiecf 12-bit data. The consequence of this is that there dots not exist a true backup (default) mode when Input data is not truly 12 bits/sample. For example, instead of using an 8 bits/sample PSIbu (e.g., as dictated by ? able 1), a JPL second-generation coder would instead use some intermediate split-sample mode beyond PSI1 ,6. In the tare eventuality that PSIbu Is needed, this" shortcoming would incur a penalty of over 1 bit/sample in the block In which It occurs, Each MRC design uses a true backup.

By (22) and (24), PSIbu in (7) is always assigned the identifier id = N-1. Thus an N=11 option coder would assign the binary four-tuple identifier "101O' (for ten). JPL's N=11 designs Instead assign the "all-ones four-tuple" to the PSIbu Identifier. Similarly, and more generally, the MRC designs assign the "all-ones four tuple" 10 the PSIbu Identifier for N>8 and the "all-ones three-tuple" when N≤8.

Another distinction between the JPL and MRC designs lies in the method for determining which code option to use. The JPL coders basically use the approach Illustrated In Table 1 where decisions are based on F_0 alone. (This is generally the most desirable technique for software implementations because of the minimal computation requirements.) MRC coders instead make decisions based on the exact bit count for each option (requiring the calculation of each F_k in (28)). The difference in average performance between the two methods has been shown to be of no practical significance.

By (21),(13) and (1 4), the form of a coded block k

$$ID(id) * PSI1\,(\tilde{M}n,k) * \tilde{L}k$$

Here, the JPL and MRC approaches diverge slightly, The MRC format follows the definition of L_k given by (11) and (12). However, the JPL format splits $\tilde{L}_k$ further into k subsequences, each containing all the ℓsbs from each sample which are of the same significance, While this provided a simplicity In the JPL coder design, it incurs a penalty on the operations required by a decoder.

Some additional differences are noted in a later section on *pre-processing*. For specific details on these implementations, consult Refs. 6-16,

More Generality

The adaptive coder we have designated as PSIss is actually a subset of the mom general coders in Ref. 15(PSI14 and P3114,K), The latter coder definitions

1) Permit λ to be zero or negative, thus allowing for additional "low-entropy" code options that provide improved performance below 1.S bits/sample. Ref. 15 and earner papers provide various low-entropy code options which can be incorporated Into this structure or can stand alone as separate" approaches. Yeh has provided a computationally simple

algorithm for incorporation in the second-generation MRC design.[14]

2) Provide a second parameter, K, for adjusting the Dynamic Range. K basically specifies a fixed K-bit pre-split of data samples, before or after a pre-processor.

3) An extended coding structure that allows these same algorithms to also be applied to the representation of code Identifiers. This reduces the overhead penalty when operating many-optioned coders on fairly stationary data.

IV. The Pre-Processor

The entropy coder, PSIss, as described in the previous section, was designed to efficiently represent (pre-processed) Standard Data Sources. PSIss doesn't need to know which pre-processor was used to produce its input. However, for an extremely broad set of real problems, the general pre-processing function of Fig. 1 can be replaced by the more specific Basic Predictive Pre-processor in Fig 4. It is shown imbedded within the complete coding module for your convenience.

Standard Predictor

The first part of this pre-processor is a very simple predictor consisting of a single sample-delay element. With x_i as the i^{th} sample in $\tilde{X}^n$, this delay element "predicts" that x_i equals the previous sample:

$$\hat{x}_i = x_{i-1} \tag{35}$$

The previous sample could be the last sample from a previous block when coding multiple blocks. It is assumed that the sample delay is always initialized with a prediction. But note also at this time that the module's design allows for an External Prediction to be supplied in place of this simple one-dimensional form.

The difference between any sample and its prediction produces the error signal

$$\Delta_i = x_i - \hat{x}_i \tag{36}$$

and the block of J error values

$$\tilde{\Delta} = \Delta_1 \Delta_2 \ldots \Delta_J \tag{37}$$

As a new data source, $\tilde{\Delta}$ sequences tend to be uncorrelated and display a unimodal distribution around zero (when data values are not near the boundaries of its dynamic range). That is

$$\Pr[\Delta_i = 0] \geq \Pr[\Delta_i = -1] \geq \Pr[\Delta_i = +1] \geq \Pr[\Delta_i = -2]^2 \ldots \tag{38}$$

The Mapper (into Standard Source)

When the latter condition in (36) is true, the following function will map each Δ_i into a corresponding Standard Source δ_i such that

$$p_0 \geq p_1 \geq p_2 \geq p_3 \geq \ldots$$

Additionally, it will assure that an n-bit/sample x_i from $\tilde{X}^n$ produces an n-bit/sample δ_i. Further, the desired probability ordering of the δ_i is more closely approximated when x_i values are near 0 or $x_{max} = 2^n - 1$.

The Mapper, used in the MRC design, is defined by

$$\delta_i = \begin{cases} 2\Delta_i & 0 \leq \Delta_i \leq \theta \\ 2|\Delta_i| - 1 & -\theta \leq \Delta_i < 0 \\ \theta + |\Delta_i| & \text{Otherwise} \end{cases} \tag{39}$$

---

² An equally valid assumption is

$$\Pr[\Delta_i = 0] \geq \Pr[\Delta_i = +1] \geq \Pr[\Delta_i = -1] \geq \Pr[\Delta_i = +2] \geq \ldots$$

which is the basis of the JPL VLSI chips. It leads to a mapping function which looks very similar to (39) and (40). Reference 15 shows that a "decoder" using one mapping function could be used to decode data that had been generated using the other mapping function.

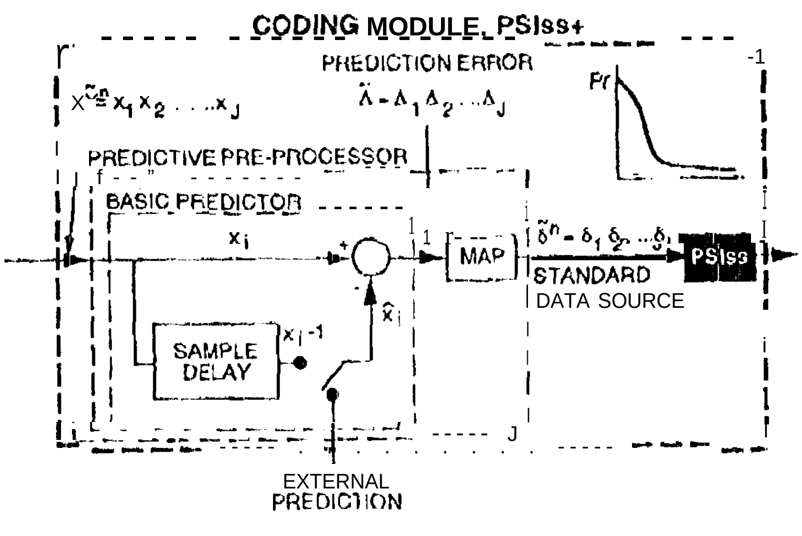

Fig. 4. Basic Predictive Preprocessor Within Coding Module, PSIss+

where

$$\theta = \min(\hat{x}_i, x_{max} - \hat{x}_i)$$
$$x_{max} = 2^n - 1 \quad (40)$$

A Further Benefit- Suppose the pre-processor in Fig 4 is set to receive an external prediction, and fix that prediction to

$$\hat{x}_i = 0 \quad (41)$$

Then, tracing through (39) and (40), one finds that

$$\delta_i = x_i \quad (42)$$

That is, the input to the pre-processor, $\tilde{X}$, is passed directly through unchanged, becoming as $\tilde{\delta}$ This means that the separate desired external input line in Fig. 1 (to allow the pre-processor to be skipped) can be omitted.

The Ideal Case

If one assumes that the distribution of $\tilde{\Delta}$ samples in (38) fits the Laplacian form, then the code equivalence result in (16) can be proven.[17,18] That is,

the simple PSI1,k codes are equivalent to Huffman Codes for Laplacian distributions of prediction errors. (43)

Reference Sample

Most of those applications which make use of the built-in Predictive Pre-processor will occasional need to incorporate a "Reference Sample" along with the coded prediction errors (e.g., at the start of an image line) Each of the VLSI implementations incorporate an optional feature to extract such a Reference Sample from an incoming data stream. The JPL and MRC approaches to formatting this Reference Sample are distinctly different. Consult Refs. 10, 11, 14 and 15.

V. Performance Comparisons

References 12 and 18 compare the performance of a complete PSI14+ coding module with the well-known Lempel-Ziv, Adaptive Huffman and Arithmetic coding algorithms. More recently, Ref. 18 compares PSI14+ with a two-pass JPEG noiseless coder.

Acknowledgments

The Research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and the Goddard Space Flight Center, under a contract with the National Aeronautic and Space Administration.

References

1) R. F. Rice and J. R. Plaunt, "Adaptive Variable Length Coding for Efficient Compression of Spacecraft Television Data," IEEE Trans. on Communication Technology, Vol. COM-19, Part I, Dec. 1971, pp. 889-897.

2) R. F. Rice, "Some Practical Universal Noiseless Coding Techniques," JPL Publication 79-22, Jet Propulsion Laboratory, Pasadena, California, March 15, 1979.

3) R. F. Rice, "Practical Universal Noiseless Coding," 1979 SPIE Symposium Proceedings, Vol. 207, San Diego, California, August 1979.

4) R. F. Rice and J. Lee, 'Some Practical Universal Noiseless Coding Techniques," Part II, JPL Publication 83-17, Jet Propulsion Laboratory, Pasadena, California, March 1983.

5) R. F. Rice, "Universal Noiseless Coding Techniques (Electronic Slideshows)," NASA Workshop on Data Compression, Snowbird, Utah, May 1908

6) J. Lee et al., "A Very High Speed Noiseless Data Compression Chip for Space Imaging Applications," Proceedings IEEE Data Compression Conference, Snowbird, Utah, April 1991.

7) J. Venbrux and N. Liu, "Lossless Image Compression Chip Set," Proceedings of Northcon, Seattle, Washington, Oct. 9-11, 1990, pp. 145-150.

8) J. Venbrux and N. Liu, "A Very High Speed Lossless Compression/Decompression Chip Set, " JPL Publication 91.13, Jet Propulsion Laboratory, Pasadena, California, July 15, 1991.

9) J. Venbrux and Pen-Shu Yeh, 'A VLSI Chip Set for High-Speed Lossless Data Compression," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 2, No 4, December 1992.

10) .... "Specification for first generation MRC lossless coder chip set," Advanced Hardware Architectures, Inc., Moscow, Idaho, 1991,

11) J. Bowers, "Chip Specification for the Cassini Data, Compression Chip," JPL Internal Memorandum, Jet Propulsion Laboratory, Pasadena, California, December 1, 1992.

12) J. Venbrux et al., "VLSI Chip Set Development for Lossless Data Compression," Proceedings of the AIAA Computing in Aerospace 9 Conference, San Diego, California, October 19-21, 1993.

13) J. Miko et al., "A High Speed Lossless Data Compression System for Space Applications," Proceedings of the AIAA Computing in Aerospace 9 Conference, San Diego, California.

14) J. Venbrux, "Universal Source Encoder for Space (USES) Specification," Internal Document, Microelectronics Research Laboratory, University of New Mexico, Albuquerque, New Mexico.

15) R.F. Rice, "Practical Universal Noiseless Coding Technique s," Part III, JPl Publication 91.3. Jet Propulsion Laboratory, Pasadena, California, November 15, 1991.

16) R, F. Rice, et al., "Algorithms for a Very High Speed Universal Noiseless Coding Module," JPL Publication 91-1, Jet Propulsion Laboratory, Pasadena, California, February 15, 1991.

1/) Pen-Shu Yeh et al., 'On the Optimally of Code Options for a Universal Noiseless Coder," JPL Publication 91-2, Jet Propulsion Laboratory, Pasadena, California, February 15, 1991.

10) Pen-Shu Yeh et al., "On the Optimality of a Universal Lossless Coder," Proceedings of the AIAA Computing in Aerospace 9 Conference, San Diego, California, October 19-21, 1993.